

GROUPS WITH CONTEXT-FREE CO-WORD PROBLEM AND EMBEDDINGS INTO THOMPSON'S GROUP V

ROSE BERNIS-ZIEVE, DANA FRY, JOHNNY GILLINGS, HANNAH HOGANSON,
AND HEATHER MATHEWS

ABSTRACT. Let G be a finitely generated group, and let Σ be a finite subset that generates G as a monoid. The *word problem of G with respect to Σ* consists of all words in the free monoid Σ^* that are equal to the identity in G . The *co-word problem of G with respect to Σ* is the complement in Σ^* of the word problem. We say that a group G is *coCF* if its co-word problem with respect to some (equivalently, any) finite generating set Σ is a context-free language.

We describe a generalized Thompson group $V_{(G,\theta)}$ for each finite group G and homomorphism $\theta: G \rightarrow G$. Our group is constructed using the cloning systems introduced by Witzel and Zaremsky. We prove that $V_{(G,\theta)}$ is *coCF* for any homomorphism θ and finite group G by constructing a pushdown automaton and showing that the co-word problem of $V_{(G,\theta)}$ is the cyclic shift of the language accepted by our automaton.

A version of a conjecture due to Lehnert says that a group has context-free co-word problem exactly if it is a finitely generated subgroup of V . The groups $V_{(G,\theta)}$ where θ is not the identity homomorphism do not appear to have obvious embeddings into V , and may therefore be considered possible counterexamples to the conjecture.

Demonstrative subgroups of V , which were introduced by Bleak and Salazar-Diaz, can be used to construct embeddings of certain wreath products and amalgamated free products into V . We extend the class of known finitely generated demonstrative subgroups of V to include all virtually cyclic groups.

1. INTRODUCTION

Let G be a group and let $\Sigma \subseteq G$ be a finite set that generates G . The *word problem of G with respect to the free monoid Σ^** is the set of all words in Σ^* that are equivalent to the identity in G . The *co-word problem of G with respect to Σ^** is the complement of the word problem. Both the word problem and the co-word problem of G are languages. The Chomsky Hierarchy [3] states that the set of regular languages is a subset of context-free languages, the set of context-free languages is a subset of context sensitive languages, and the set of context sensitive languages is a subset of recursive languages. We will focus in particular on context-free languages. A language is *context-free* if it is accepted by a pushdown automaton. If the co-word problem of G is a context-free language, then we say G is *coCF*. This property does not depend on the choice of monoid generating set. The class of *coCF* groups was first studied by Holt, Rees, Röver, and Thomas [6]. They showed that the class is closed under taking finite direct products, taking restricted standard wreath products with virtually free top groups, and passing to finitely generated subgroups and finite index overgroups.

2010 *Mathematics Subject Classification.* 20F10, 20E06.

Key words and phrases. context-free language, pushdown automaton, Thompson's groups.

One group of particular interest is Thompson's group V , which is an infinite but finitely presented simple group. Lehnert and Schweitzer demonstrate that Thompson's group V is coCF . This group is of interest to us because of the conjecture, formulated by Lehnert and revised by Bleak, Matucci, and Neunhöffer [1], that any group with context-free co-word problem embeds in V , i.e.,

Conjecture 1.1. Thompson's group V is a universal coCF group.

In this paper we prove two classes of results, one related to embeddings into V , and the other offering a potential counterexample to Conjecture 1.1.

Bleak and Salazar-Diaz [2] define the class of demonstrative subgroups of V and use this class to produce embeddings of free products and wreath products into V . They also show that the class of groups that embed into V is closed under taking finite index overgroups. Their proof of the latter fact appeals to results of Kaloujnine and Krasner [8]. Here, we use induced actions to give a direct proof. Our argument shows, moreover, that if the original embedding is demonstrative, then so is the embedding of the finite index overgroup.

A theorem of [2] says that \mathbb{Z} is a demonstrative subgroup of V . The results sketched above prove that all virtually cyclic groups are demonstrative, and it appears that these are the only known finitely generated demonstrative subgroups. If V is a universal coCF group, then it should be possible, by the results of Holt, Rees, Röver, and Thomas [6], to find an embedding of $G \wr F_2$ into V , where G is coCF and F_2 is the free group on two generators. The easiest way to find such an embedding would be to show that F_2 has a demonstrative embedding into V . We are thus led to ask:

Question 1.2. *Does there exist a demonstrative embedding of F_2 into V ?*

Our class of potential counterexamples to Conjecture 1.1 comes from the cloning systems of Witzel and Zaremsky [11]. We look at a specific group $V_{(G,\theta)}$ that arises from their family of groups equipped with a cloning system. We define a surjective homomorphism Φ from $V_{(G,\theta)} \rightarrow V$, which implies that $V_{(G,\theta)}$ acts on the Cantor set. However, by our construction, $V_{(G,\theta)}$ seems to have no obvious faithful actions on the Cantor set when θ is not the identity homomorphism.

In our main result, we prove that $V_{(G,\theta)}$ is coCF for all pairs of θ and finite G . We begin by detailing a construction of a pushdown automaton and we show that the co-word problem is equivalent to the cyclic shift of the language accepted by the automaton, therefore proving that the co-word problem is context-free.

We briefly outline the paper. Section 2 provides the necessary background for the reader to understand the concepts discussed in the two following sections. In Section 3, we give our proofs and examples of all ideas related to demonstrative subgroups. Finally, in Section 4 we prove the main result of our paper.

2. BACKGROUND

2.1. Pushdown Automata.

Definition 2.1. Let Σ be a finite set, called an *alphabet*. We call elements of the alphabet *symbols*. The *free monoid* on Σ , denoted Σ^* , is the set of all finite strings of symbols from Σ . This includes the empty string, which we denote ϵ . The operation is concatenation. An element of the free monoid is a *word*. A subset of the free monoid is a *language*.

Example 2.2. Let $\Sigma = \{0, 1\}$. The free monoid Σ^* contains all finite concatenations of 0 and 1 in any order. An example word is 01101.

Definition 2.3. Let G be a group. A *finite monoid generating set* is a finite alphabet Σ with a surjective monoid homomorphism $\Phi : \Sigma^* \rightarrow G$. The *word problem* of a group G (with respect to Σ), denoted $WP_\Sigma(G)$, is the kernel of Φ . The complement of the word problem is the *co-word problem*, denoted $CoWP_\Sigma(G)$.

Definition 2.4. Let Σ, Γ be alphabets and let $\#$ be an element of Γ . A *pushdown automaton* [3] with stack alphabet Γ and input alphabet Σ is defined as a directed graph with a finite set of vertices V , a finite set of *transitions* (directed edges) δ , an initial state $v_0 \in V$, and a set of *terminal states* $T \subseteq V$.

A transition is labeled by an ordered triplet $(w_1, w_2, w_3) \in (\Sigma \cup \{\epsilon\}) \times \Gamma^* \times \Gamma^*$. When *following* a transition, the pushdown automaton (PDA) reads and deletes w_1 from its input tape, reads and deletes w_2 from its memory stack (shortened to stack for the duration of this paper), and writes w_3 on its stack. If w_1, w_2 , or w_3 equals ϵ , the automaton does not execute the action associated with that coordinate. We only consider *generalized* PDA, which (as described above) can add and delete multiple letters on its stack at a time. A PDA accepts languages either by terminal state, or by empty stack. This must be specified upon creation of the automaton. See Definitions 2.6 and 2.7.

Definition 2.5. ([5], Definition 2.6) Let P be a pushdown automaton. We describe a class of directed paths in P , called the *valid paths*, by induction on length. The path of length 0 starting at the initial vertex $v_0 \in P$ is valid; its stack value is $\# \in \Gamma^*$. Let $t_1 \dots t_n (n \geq 0)$ be a valid path in P , where t_1 is the transition crossed first. Let t_{n+1} be a transition whose initial vertex is the terminal vertex of t_n ; we suppose that the label of t_{n+1} is (s, w_1, w_2) . The path $t_1 \dots t_n t_{n+1}$ is also valid, provided that the stack value of $t_1 \dots t_n$ has w_1 as a prefix; that is, if the stack value of $t_1 \dots t_n$ has the form $w_1 w' \in \Gamma^*$ for some $w' \in \Gamma^*$. We say that the edge t_{n+1} is a *valid transition*. The stack value of $t_1 \dots t_n t_{n+1}$ is then $w_2 w'$. We let $val(p)$ denote the stack value of a valid path p .

The *label* of a valid path $t_1 \dots t_n$ is $s_n \dots s_1$, where s_i is the first coordinate of the label for t_i (an element of Σ , or the empty string). The label of a valid path p will be denoted $\ell(p)$.

Definition 2.6. Let p be a valid path of a pushdown automaton P . We say that p is a *successful path* when

- (1) $val(p) = \epsilon$ if P accepts by empty stack, or
- (2) The terminal vertex of p is in T if P accepts by terminal state.

Definition 2.7. Let P be a PDA. The language *accepted by* P , denoted \mathcal{L}_P , is

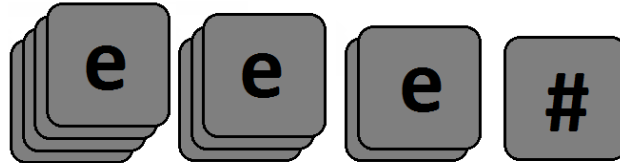


FIGURE 1. The stack of a PDA. As an element is read from the top, the next element appears as the new top if nothing is written to the stack.

$\mathcal{L}_P = \{w \in \Sigma^* \mid w = \ell(p) \text{ for some successful path } p\}$.

Definition 2.8. A subset of the free monoid Σ^* is called a *(non-deterministic) context-free language* if it is \mathcal{L}_P for some pushdown automaton P .

Let P be a PDA. We operate P in the following way:

- (1) A word \hat{w} is placed on the input tape and read symbol by symbol. By our convention, P reads the input tape from right to left.
- (2) Next, P follows valid transitions non-deterministically (i.e., by choosing them) until $\hat{w} = \ell(p)$ for some successful path p . Throughout this process, the leftmost symbol on the stack is considered to be in the top position.
- (3) If some successful path p exists, P accepts \hat{w} .

2.2. Thompson's Group V .

Definition 2.9. Let $X = \{0, 1\}$ and consider X^* . As in [2], we define an infinite rooted tree, \mathcal{T}_2 , as follows:

The set of nodes for \mathcal{T}_2 is X^* . For $u, v \in X^*$, there exists an edge from u to v if $ux = v$ for some $x \in X$.

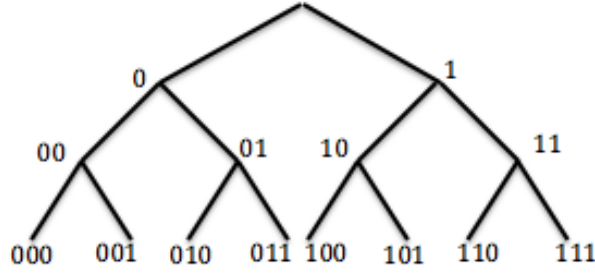


FIGURE 2. The top portion of the infinite binary tree \mathcal{T}_2 with some of its nodes labeled.

Definition 2.10. An *infinite path* \mathcal{T}_2 is an infinite string of 0s and 1s. $Ends(\mathcal{T}_2)$ is the collection of all such infinite paths.

We say that $u \in X^*$ is a *prefix* of $\omega \in Ends(\mathcal{T}_2)$ if there is $\hat{\omega} \in Ends(\mathcal{T}_2)$ such that $\omega = u\hat{\omega}$. For $u \in X^*$, we let $u* = \{\omega \in Ends(\mathcal{T}_2) \mid u \text{ is a prefix of } \omega\}$.

Define $d : Ends(\mathcal{T}_2) \times Ends(\mathcal{T}_2) \rightarrow \mathbb{R}$ by $d(\zeta_1, \zeta_2) = e^{-l}$, where $\zeta_1, \zeta_2 \in Ends(\mathcal{T}_2)$ and l is the length of the longest prefix shared by ζ_1 and ζ_2 . The function d is a metric on $Ends(\mathcal{T}_2)$.

For $w \in Ends(\mathcal{T}_2)$, let $B_r(w) = \{\zeta \in Ends(\mathcal{T}_2) \mid d(\zeta, w) \leq r\}$. Then $B_r(w)$ is the *metric ball around w with radius r* . It can be shown that each metric ball in $Ends(\mathcal{T}_2)$ takes the form $u*$, for some $u \in X^*$.

We note that $Ends(\mathcal{T}_2)$ is a Cantor set. Thompson's group V acts as self-homeomorphisms on this Cantor set, and each element of V can be represented by a binary tree pair. Furthermore, the leaves of the trees can be represented in binary code where a branch to the left is denoted by "0" and a branch to the right by "1."

The group V is generated by the maps A, B, C and π_0 [4]. We define the generators of V by the prefix changes they represent, which are equivalent to the tree diagrams in Figure 3. For instance, if $\omega \in \text{Ends}(\mathcal{T}_2)$ has the form $\omega = 0\hat{\omega}$, for some $\hat{\omega} \in \text{Ends}(\mathcal{T}_2)$, then $A(\omega) = 00\hat{\omega}$.

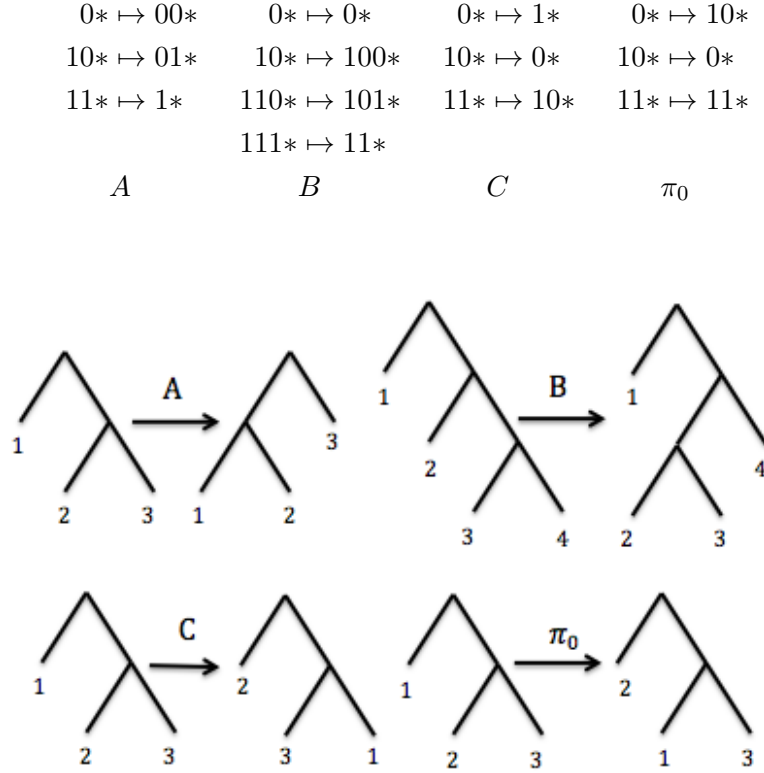


FIGURE 3. Elements A , B , C , and π_0 of V represented as tree pairs.

2.3. Generalized Thompson Groups $V_{(G,\theta)}$.

Definition 2.11. [11] The *forest monoid*, \mathcal{F} , consists of all sequences of ordered, rooted, binary trees $(T_i)_{i \in \mathbb{N}}$, where all but finitely many trees are trivial (i.e., consist only of the root). For two elements $E_1, E_2 \in \mathcal{F}$, their product, $E_1 E_2$, is obtained by attaching the i th leaf of E_1 with the i th root of E_2 .

Given a finite group, G , define

$$H = S_\infty \ltimes_\phi (\oplus_{i=1}^\infty G)$$

where $\phi : S_\infty \rightarrow \text{Aut}(\oplus_{i=1}^\infty G)$ by $\phi(\sigma)(g_1, \dots, g_k, \dots) = (g_{\sigma^{-1}(1)}, \dots, g_{\sigma^{-1}(k)}, \dots)$ for $\sigma \in S_\infty$.

Multiplication of group elements $(\sigma_1, (g_1, \dots, g_k, \dots)), (\sigma_2, (g'_1, \dots, g'_k, \dots)) \in H$ works as follows:

$$(\sigma_1, (g_1, \dots, g_k, \dots))(\sigma_2, (g'_1, \dots, g'_k, \dots)) = (\sigma_1 \sigma_2, \phi(\sigma_2)(g_1, \dots, g_k, \dots)(g'_1, \dots, g'_k))$$

$$= (\sigma_1 \sigma_2, (g_{\sigma_2^{-1}(1)} g'_1, \dots, g_{\sigma_2^{-1}(k)} g'_k, \dots))$$

Our group $V_{(G,\theta)}$ arises from the cloning system construction in [11]. A *cloning system* consists of a group H , a homomorphism $\rho : H \rightarrow S_\omega$, and a collection of cloning maps $\{\kappa_k \mid k \in \mathbb{N}\}$. A cloning system with these three elements that satisfies conditions *CS1*, *CS2*, *CS3* of Proposition 2.7 [11] defines a BZS product, $\mathcal{F} \bowtie H$. We do not go into detail on the BZS product; for a full discussion see [11].

Elements of $V_{(G,\theta)}$ are ordered pairs of elements from a subgroup of the BZS product defined by the following cloning system:

$$H = S_\infty \ltimes_\phi (\oplus_{i=1}^\infty G),$$

$$\rho(\sigma, (g_1, \dots, g_k)) = \sigma,$$

where κ_k acts on the right by:

$$(\sigma, (g_1, \dots, g_k, g_{k+1}, \dots)) \kappa_k = (\sigma \zeta, (g_1, \dots, g_k, \theta(g_k), g_{k+1}, \dots))$$

where ζ is the cloning map for the symmetric group defined in Example 2.9. of [11], and θ is an arbitrary homomorphism from $G \rightarrow G$.

We can think of elements of $V_{(G,\theta)}$ as equivalence classes of tree pairs much as in Thompson's group V . The difference is that tree pairs in $V_{(G,\theta)}$ have group elements from G attached to their leaves. A tree pair (a, b) can be modified within its equivalence class by adding canceling carets or canceling group elements.

Canceling carets are added to corresponding leaves of the domain and range trees just as they would be in Thompson's group V , unless there is a group element g on the leaf, in which case we put a g on the left branch of the new caret and $\theta(g)$ on the right branch. Canceling group elements are added to corresponding leaves of the domain and range trees and are combined, using the group operation, with any group element already on the leaves. We can multiply two elements $(a, b), (c, d) \in V_{(G,\theta)}$, by choosing equivalent tree pairs (a', b') and (c', d') where $b' = c'$. Then $(a, b)(c, d) = (a', b')(c', d') = (a', d')$. (Note that, in these tree pairs, the second coordinate corresponds to the domain, and the first to the range.)

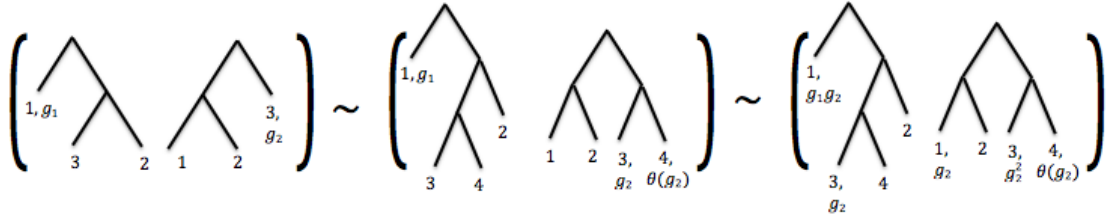


FIGURE 4. Three equivalent tree pairs, the second is obtained by adding a canceling caret and the third by canceling group elements.

Note that any element $(a, b) \in V_{(G,\theta)}$ can be expressed with no group elements in the domain tree by adding canceling group elements.

If $\{g_1, \dots, g_n\}$ is a generating set for G , then $\{A, B, C, \pi_0\} \cup \{g_{ja}, g_{jb}, g_{jc}, g_{jd}, g_{je} \mid 1 \leq j \leq n\}$ is a generating set for $V_{G,\theta}$, where $g_{ja}, g_{jb}, g_{jc}, g_{jd}, g_{je}$ are defined as in Figure 6.

Remark 2.12. When $\theta = id_G$, the identity homomorphism, $V_{(G,\theta)}$ embeds in V . Consider $V_{(G,\theta)}$ where θ is the identity homomorphism, $G = \{g_1, \dots, g_n\}$, and $(a, b) \in V_{(G,\theta)}$. We

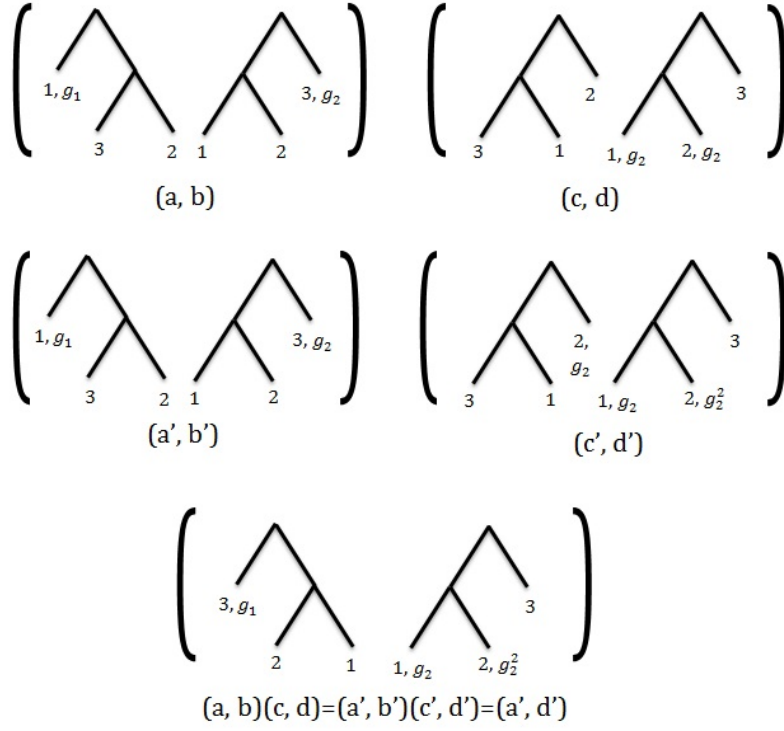
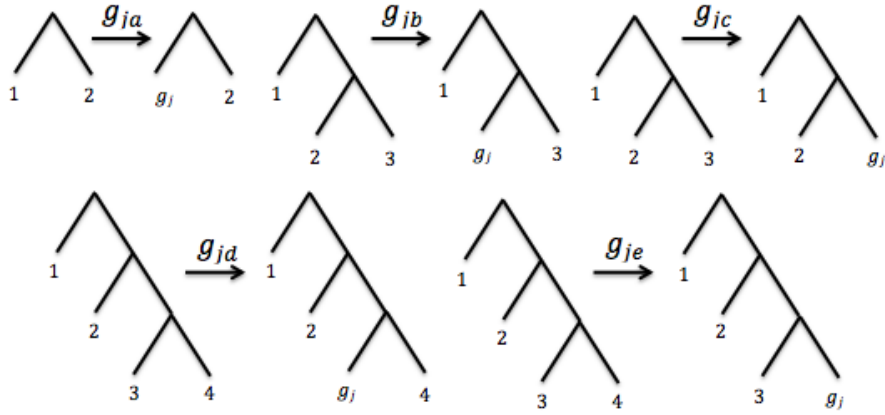


FIGURE 5. Multiplication of two group elements (a,b) and (c,d).

FIGURE 6. Additional generators of $V_{(G, \theta)}$.

assume further that the domain tree b has no group elements on its leaves. Choose a partition $W = \{w_1^*, \dots, w_n^*\}$ where w_i will be the prefix associated with group element g_i . Consider the j th leaf of the tree pair, $b_j^* \rightarrow a_j g_j^*$. If $g_i g_j = g_k$, then $w_i b_j^*$ is mapped

to $w_k b_j *$. This assignment encodes each $(a, b) \in V_{(G, \theta)}$ as an element of V in an injective fashion. However, when $\theta \neq id_g$, the same method of embedding fails.

3. DEMONSTRATIVE GROUPS

Definition 3.1. [2] Suppose a group G acts by homeomorphisms on a topological space X . For a group $H \leq G$, the action of H in G is *demonstrative* if and only if there exists an open set $U \subset X$ such that for all $h_1, h_2 \in G$, $h_1 U \cap h_2 U \neq \emptyset$ if and only if $h_1 = h_2$. The set U is called a *demonstration set*.

For this discussion, let $G = V$ and $X = Ends(\mathcal{T}_2)$. In this case, if U (as above) is a metric ball (i.e. $U = w*$ for some $w* \subseteq Ends(\mathcal{T}_2)$), then we refer to U as a *demonstration node*.

Definition 3.2. Let $H \leq G$ and let H act on a topological space S . We define $G \times_H S$ to be $\{(g, s) : g \in G, s \in S\} / \sim$, where $(gh, s) \sim (g, h \cdot s)$.

The *induced action* of G on S is $*$: $G \times (G \times_H S) \rightarrow G \times_H S$ defined by $g_1 * (g_2, s) = (g_1 g_2, s)$.

Theorem 3.3. If $H \leq G$ where $[G : H] = m$, for some $m \in \mathbb{N}$, then G embeds in V . Moreover, if H embeds as a demonstrative subgroup in V , then G embeds as a demonstrative subgroup of V .

Proof. Assume $H \leq G$. Choose a left transversal $T = \{t_1, t_2, \dots, t_m\}$ for H in G with $t_1 = 1$. We can induce an action of G on $G \times_H Ends(\mathcal{T}_2)$ by:

$$g \cdot (t_i, x) = (gt_i, x), \text{ for } x \in Ends(\mathcal{T}_2)$$

We know we can write gt_i as $t_j h$ for unique $t_j \in T$ and $h \in H$. So,

$$(gt_i, x) = (t_j h, x) = (t_j, h \cdot x)$$

Now, we can embed $G \times_H Ends(\mathcal{T}_2) \hookrightarrow Ends(\mathcal{T}_2)$ by choosing a set $W = \{w_1, \dots, w_m\}$ such that $\{w_1*, w_2*, \dots, w_m*\}$ is a partition of $Ends(\mathcal{T}_2)$, and defining an injective function $\phi : T \rightarrow W$ by $\phi(t_i) = w_i$. Now define $\Phi : G \times_H Ends(\mathcal{T}_2) \rightarrow Ends(\mathcal{T}_2)$ by $\Phi((t_i, x)) = \phi(t_i)x = w_i x$.

It can be easily checked that

$$g \cdot (w_i x) = g \cdot \Phi(t_i, x) = \Phi(gt_i, x) = \Phi(t_j, h \cdot x) = w_j h(x)$$

is a group action of G on $Ends(\mathcal{T}_2)$. Additionally, because elements of H act as elements of V and $[G : H] < \infty$, so do elements of G . Therefore, G embeds in V .

Now, assume H has a demonstrative embedding in V with demonstration node $a_1 a_2 \dots a_n *$ for $a_i \in \{0, 1\}$.

We will show that $w_i a_1 \dots a_n *$ is a demonstration node for G . We compute the action of each of $g, g' \in G$ on $w_i a_1 \dots a_n *$:

$$g \cdot w_i a_1 \dots a_n * = w_j h(a_1 \dots a_n *)$$

$$g' \cdot w_i a_1 \dots a_n * = w_k h'(a_1 \dots a_n *)$$

Since w_1*, \dots, w_m* partition $Ends(\mathcal{T}_2)$, if $w_j * \cap w_k * \neq \emptyset$ then we have $j = k$. Since H is a demonstrative subgroup of G with demonstration node $a_1 a_2 \dots a_n *$, $h(a_1 \dots a_n *) \cap h'(a_1 \dots a_n *) \neq \emptyset$ if and only if $h = h'$. Thus, $w_j h(a_1 \dots a_n *) \cap w_k h'(a_1 \dots a_n *) \neq \emptyset$ if and only if $j = k$ and $h = h'$, in other words, if and only if $g = g'$. Thus, G is demonstrative in V with demonstration node $w_i a_1 \dots a_n *$ for any $i \in \{1, \dots, m\}$. \square

4. MAIN RESULT

Lemma 4.1. *Let $\Sigma = \{A, B, C, \bar{A}, \bar{B}, \pi_0, g_{ij}\}$, where $i \in \{1, \dots, n\}$, and $j \in \{a, b, c, d, e\}$. Define $\Phi : \Sigma^* \rightarrow V$ by the homomorphism*

$$\begin{aligned} \Phi : A &\mapsto A \\ \bar{A} &\mapsto A^{-1} \\ B &\mapsto B \\ \bar{B} &\mapsto B^{-1} \\ C &\mapsto C \\ \pi_0 &\mapsto \pi_0 \\ g_{ij} &\mapsto 1_V \end{aligned}$$

If $w = b_1 \cdots b_m \in \Sigma^$ satisfies $\Phi(w) \neq 1$, then there is a cyclic permutation $b_j \cdots b_m b_1 \cdots b_{j-1}$ and some $B \in \{a_1 a_2 a_3^* : a_i \in \{0, 1\}\}$ that satisfy $b_j \cdots b_m b_1 \cdots b_{j-1}(B) \cap B \neq \emptyset$. (Here the action of a word $w \in \Sigma^*$ on the ball B is determined by the rule $w(B) = \Phi(w)(B)$.)*

Proof. Consider $w = b_1 \cdots b_m \in \Sigma^*$. If $\Phi(w) \neq 1$, then $\Phi(w) \in \text{CoWP}(V)$. We know $\{a_1 a_2 a_3^* : a_i \in \{0, 1\}\}$ is a test partition for V [5]. So, there is some cyclic permutation $\Phi(w)'$ of $\Phi(w)$ and some $B \in \{a_1 a_2 a_3^* : a_i \in \{0, 1\}\}$ such that $\Phi(w)'(B) \cap B \neq \emptyset$. Since Φ takes all the generators g_{ij} to 1, Φ will preserve the shape of any tree pair. Thus, if $\Phi(w)'$ is such that $\Phi(w)'(B) \cap B \neq \emptyset$, then $w'(B) \cap B \neq \emptyset$ where w' is some cyclic shift $b_j \cdots b_m b_1 \cdots b_{j-1}$ of w . \square

Definition 4.2. Let \mathcal{L} be a language. The *cyclic shift* of \mathcal{L} , denoted \mathcal{L}° , is

$$\mathcal{L}^\circ = \{w_2 w_1 \in \Sigma^* \mid w_1 w_2 \in \mathcal{L}, w_1, w_2 \in \Sigma^*\}.$$

A *cyclic permutation* w' of a word $w = w_1 w_2$ is $w' = w_2 w_1$. Note that the class of context-free languages is closed under cyclic shifts [10].

Theorem 4.3. *The group $V_{(G, \theta)}$ is coCF.*

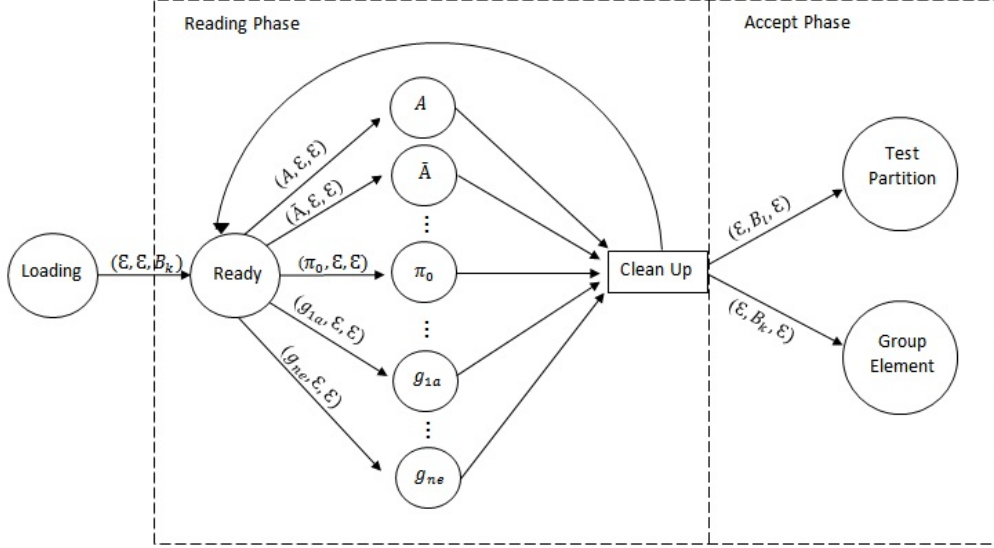
Proof. We design an automaton P to accept by empty stack, with stack alphabet $\Gamma = \{0, 1, g \mid g \in G \setminus \{1_G\}\}$ and input alphabet $\Sigma = \{A, B, C, \bar{A}, \bar{B}, \pi_0, g_{ij}\}$, where $i \in \{1, \dots, n\}$, and $j \in \{a, b, c, d, e\}$.

We define

$$\mathcal{L}_{B_i} = \{w \in \Sigma^* \mid w(B_i) \cap B_j \neq \emptyset \text{ for some } j \neq i\}.$$

We let \mathcal{L}_G be the set of words w in Σ^* such that there is a tree pair representative for $w \in V_{(G, \theta)}$ with no group elements written on the leaves of the domain tree, and at least one non-trivial $g \in G$ written on a leaf of the range tree.

We design P such that $\mathcal{L}_P = (\bigcup_{i=1}^8 \mathcal{L}_{B_i}) \cup \mathcal{L}_G$. Figure 7 outlines a portion of the automaton. Note that unlabeled arrows represent $(\epsilon, \epsilon, \epsilon)$ transitions. From the initial loading phase, there are in fact eight different arrows $(\epsilon, \epsilon, B_k)$, one for each of the $B_k \in \{000, 001, \dots, 111\}$. These lead to eight separate reading and accept phases, each as pictured in the Figure. These reading and accept phases are identical, with one exception: the labels on the arrows leading to the test partition accept state vary. For instance, in the accept phase corresponding to 000, the arrow labelled $(\epsilon, B_l, \epsilon)$ corresponds to seven different arrows, one for each $B_l = a_1 a_2 a_3$, where $a_i \in \{0, 1\}$ and not all of a_1, a_2, a_3 are 0.

FIGURE 7. Sample reading and accept phases of the automaton for $V_{(G, \theta)}$.

To start off, P enters a non-deterministic loading phase. This consists of a single state S with transitions labelled $(\epsilon, \epsilon, 0)$ and $(\epsilon, \epsilon, 1)$, both leading back to S . Here a finite string of 1's and 0's is entered non-deterministically onto the stack.

We leave the loading phase by taking a transition $(\epsilon, \epsilon, B_k)$ where

$$B_k \in \{000, 001, 010, 100, 011, 110, 101, 111\},$$

i.e. B_k is one of the 8 metric balls in the test partition.

Next, P enters the reading phase which has a single state for each generator, where the first element on the input tape is read and that generator is applied to the appropriate prefix at the top of the stack. For example, the reading phase for A would read and delete a 0, and then add 00; the reading phase for g_{2b} would read and delete 10, and then add $10g_2$.

After the reading phase, P enters the clean-up state, which consists of the pushing and combining of stack elements. This phase allows P to “clean-up” the stack so that there are at least three 0's and 1's for the next element on the input tape to successfully act on the stack. First, P “pushes” elements within the first three spots to the fourth spot on the stack. For example, as shown in Figure 4, one set of transitions will read and delete $0g0$ or $0g1$ from the stack and then add $00g$ or $01(\theta(g))$, respectively, to the stack for all $g \in G \setminus \{1_G\}$. Similar transitions can be followed if the group element is preceded by the prefix 0, 1, 01, 10, 00, or 11.

Next, P enters the combining state where group elements are rewritten as a single element of the group (in accordance with the group operation). For example, one collection of edges is able to read and delete $010(g)(g')$ and add $010(gg')$ for all $g, g' \in G \setminus \{1_G\}$. Note that if $gg' = 1_G$, then the path reads and deletes $010(g)(g')$, and writes 010. Similar paths exist when combining any two group elements preceded by any three-digit prefix.

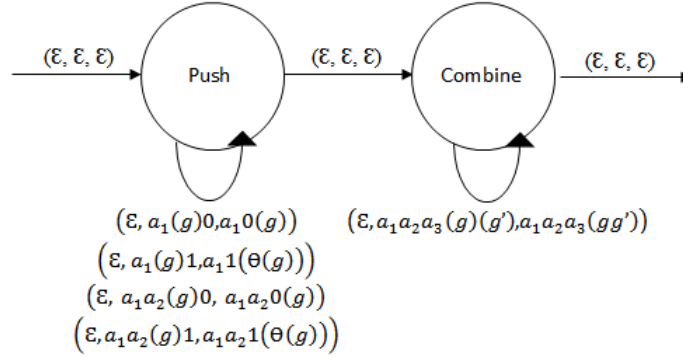


FIGURE 8. The clean-up phase. Let $a_1, a_2, a_3 \in \{0, 1\}$ and $g, g' \in G \setminus \{I_G\}$.

After exiting the clean-up phase, P reads the next element on the input tape and repeats the process of the reading phase. When the input tape is empty and P has gone through the reading and clean-up phases, P then moves onto one of two accept states.

If the word from the input tape took one metric ball B_k in the test partition to some other metric ball B_l , then the three-letter prefix describing B_l is now showing on the stack, so we can follow a path labelled $(\epsilon, B_l, \epsilon)$ to the *test partition accept state*. (Here we recall that the single arrow labelled $(\epsilon, B_l, \epsilon)$ in Figure 7 is actually seven different arrows, one for each l such that $B_l \neq B_k$.) At this point we “unload” all of the 0’s and 1’s and group elements off the stack with paths (ϵ, x, ϵ) for $x \in \{0, 1, g \mid g \in G \setminus \{I_G\}\}$. Once every stack element has been deleted, P takes the path $(\epsilon, \#, \epsilon)$ which deletes the start symbol and accepts the word. So the language accepted by the eight test partition accept states is $(\bigcup_{i=1}^8 \mathcal{L}_{B_i})$.

If the word from the input tape does not displace metric ball B_k , then we enter the *group element accept state*. Here, we delete every 0 and 1 on the stack until P arrives at a group element. The group element is then “pushed” further down the stack, and the 0 or 1 it pushes past is deleted. For example, one path is $(\epsilon, g0, g)$ for $g \in G \setminus \{I_G\}$. If the group element is followed on the stack by a second group element, then they are “combined” in a manner mimicking the previously described combining portion of the clean-up phase. This is repeated until there are no 0’s or 1’s left on the stack. At this point, if there is still a group element remaining on the stack followed by the start symbol, then they are both deleted and thus the word is accepted. However, if there is no group element on the stack then the start symbol cannot be deleted so the word is not accepted. Assuming that the address of an appropriate metric ball was written on the stack in the loading phase, there will be a group element remaining, and so the language accepted by the eight group element accept states is \mathcal{L}_G .

Thus, $\mathcal{L}_P = (\bigcup_{i=1}^8 \mathcal{L}_{B_i}) \cup \mathcal{L}_G$.

We claim that $\text{CoWP}(V_{(G, \theta)}) = (\mathcal{L}_P)^\circ$.

Let $w \in \mathcal{L}_P$, so that $w \in \mathcal{L}_G$ or $w \in \mathcal{L}_{B_i}$, for some i . If $w \in \mathcal{L}_G$, then it follows directly that $w \in \text{CoWP}(V_{(G, \theta)})$. If $w \in \mathcal{L}_{B_i}$ for some $1 \leq i \leq 8$, then $w(B_i) \cap B_j$ for $j \neq i$, so $w \in \text{CoWP}(V_{(G, \theta)})$. Therefore, $\mathcal{L}_P \subseteq \text{CoWP}(V_{(G, \theta)})$. The CoWP of a group is closed under cyclic shift, and thus $(\mathcal{L}_P)^\circ \subseteq \text{CoWP}(V_{(G, \theta)})$.

Let $w \in \text{CoWP}(V_{(G,\theta)})$. We will use the surjective homomorphism Φ from Lemma 4.1. If $w \notin \text{Ker}(\Phi)$, then $\Phi(w) \neq 1_V$. By Lemma 4.1, there is some cyclic permutation w' and some B_i such that $w'(B_i) \cap B_i \neq \emptyset$, i.e. $w' \in \mathcal{L}_{B_i}$. Therefore, $w \in (\mathcal{L}_{B_i})^\circ \subseteq (\mathcal{L}_P)^\circ$. If $w \in \text{Ker}(\Phi) \setminus \{1_{V_{(G,\theta)}}\}$, then $\Phi(w) = 1_V$, which implies that w (as a reduced tree pair) does not change any prefixes; it simply adds group elements. Therefore, $w \in \mathcal{L}_G \subseteq (\mathcal{L}_P)^\circ$.

We now have that $\text{CoWP}(V_{(G,\theta)}) = (\mathcal{L}_P)^\circ$. A language is context-free if and only if its cyclic shift is also context-free. Since \mathcal{L}_P is context-free, $\text{CoWP}(V_{(G,\theta)})$ is context-free, and $V_{(G,\theta)}$ is coCF . □

Remark 4.4. We attempted a similar method of proof with the group generated by Thompson's group V and the Grigorchuk group G , $R = \langle V, G \rangle$. Like $V_{(G,\theta)}$, elements of R can be thought of as Thompson's group V elements with Grigorchuk group elements, g attached to the leaves. However, the Grigorchuk group elements continue to act on the tree whereas the group elements in $V_{(G,\theta)}$ just sit at the end of the branches. This became a problem because it is impossible to complete the calculation of the action of g on any finite test string loaded onto an automaton. We also ran into problems because the test partitions argument used in Theorem 4.3. and for Finite Similarity Structure Groups [7] fails.

Acknowledgments. We would like to first thank our research advisor, Dr. Dan Farley, for aiding us in our research endeavors and providing us with the knowledge needed for us to be successful. We thank SUMSRI for giving us the opportunity to participate in undergraduate research in mathematics, and we thank the faculty and staff for making this program possible. In addition, we would like to express our gratitude to Miami University for providing funding, housing, and a place to do research. Finally, we thank the National Science Foundation for funding SUMSRI and making our research possible.

REFERENCES

- [1] Collin Bleak, Francesco Matucci, and Max Neunhöffer. Embeddings into thompson's group v and coCF groups. *arXiv:1312.1855*, 15 pages, 2013.
- [2] Collin Bleak and Olga Salazar-Díaz. Free products in r . thompson's group V . *Trans. Amer. Math. Soc.*, 365(11):5967–5997, 2013.
- [3] J. Glenn Brookshear. *Formal Languages, Automata, and Complexity*. Theory of Computation. The Benjamin/Cummings Publishing Company, 1989.
- [4] J. W. Cannon, W. J. Floyd, and W. R. Parry. Introductory notes on Richard Thompson's groups. *Enseign. Math. (2)*, 42(3-4):215–256, 1996.
- [5] Daniel Farley. Local similarity groups with context-free co-word problem. *arXiv:1406.4590*, 17 pages, 2014.
- [6] Derek F. Holt, Sarah Rees, Claas E. Röver, and Richard M. Thomas. Groups with context-free co-word problem. *J. London Math. Soc. (2)*, 71(3):643–657, 2005.
- [7] Bruce Hughes. Local similarities and the Haagerup property. *Groups Geom. Dyn.*, 3(2):299–315, 2009. With an appendix by Daniel S. Farley.
- [8] Marc Krasner and Léo Kaloujnine. Produit complet des groupes de permutations et problème d'extension de groupes. I. *Acta Sci. Math. Szeged*, 13:208–230, 1950.
- [9] Roger C. Lyndon and Paul E. Schupp. *Combinatorial group theory*. Classics in Mathematics. Springer-Verlag, Berlin, 2001. Reprint of the 1977 edition.
- [10] A. N. Maslov. The cyclic shift of languages. *Problemy Peredači Informacii*, 9(4):81–87, 1973.
- [11] Stefan Witzel and Matt Zaremsky. Thompson groups for systems of groups, and their finiteness properties. *arXiv:1405.5491*, 48 pages, 2014.

DEPARTMENT OF MATHEMATICS, HAMILTON COLLEGE, CLINTON, NY 13323

E-mail address: `rbernszi@hamilton.edu`

DEPARTMENT OF MATHEMATICS AND STATISTICS, MOUNT HOLYOKE COLLEGE, SOUTH HADLEY, MA
01075

E-mail address: `fry22d@mholyoke.edu`

DEPARTMENT OF MATHEMATICS, MOREHOUSE COLLEGE, ATLANTA, GA 30314

E-mail address: `j.gillingsjr@gmail.com`

DEPARTMENT OF MATHEMATICS, MIAMI UNIVERSITY, OXFORD, OH 45056

E-mail address: `hoganshl@miamioh.edu`

DEPARTMENT OF MATHEMATICS, MIAMI UNIVERSITY, OXFORD, OH 45056

E-mail address: `mathewhm@miamioh.edu`